**Bericht/Erläuterung zur Implementierung des Refactorings "Replace Primitive Literal Expression by Variable"**

**Erklärung des Refactorings:**

Bei diesem Refactoring wird ein Wert des Typs "PrimitiveLiteralExpression", der an keine Variable gebunden ist, durch eine Variable mit gleichem Wert ersetzt. Dabei ist es wichtig, dass nicht nur eine Variable mit diesem Wert angelegt wird, sondern dass auch eine Operation, die diesen Wert nutzt, an die neue Variable angepasst wird.

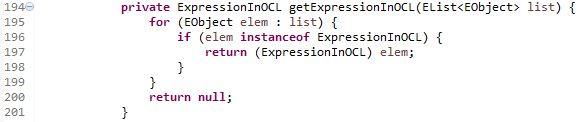
**Meine Implementierung dieses Refactorings:**

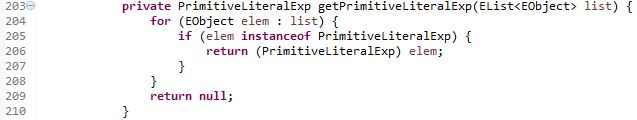
****

Zuerst wird eine Liste aller Elemente der ausgewählten "Constraint" erstellt (Z. 136-141, 144). Danach wird eine "Factory" erstellt (Z. 142), mit deren Hilfe die neuen OCL-Elemente "letExp", "var" (Variable) und "varExp" (VariableExp) generiert werden können (Z. 143, 145, 147). Über diese neuen Elemente wird eine neue Variable in die "Constraint" eingebunden. Dies wird erreicht, indem den neuen Objekten die jeweils benötigten Werte zugewiesen werden. Die Variable erhält so einen vorher vom Nutzer gewählten Namen (Z. 136-140, 146), und "varExp" erhält "var" als "referredVariable" (Z. 148). Danach wird über den Aufruf der Hilfsmethode "getExpressionInOCL(allElements)" das Element des Typs "ExpressionInOCL" gesucht (Z. 149), welches die für das Refactoring wichtige "BodyExpression" enthält. Anschließend wird die "BodyExpression" der "ExpressionInOCL" durch die neu generierte "LetExp" ersetzt, welche die alte "BodyExpression" als Wert für "in" übergeben bekommt (Z. 150-152). Danach wird über den Aufruf der Hilfsmethode "getPrimitiveLiteralExp(allElements)" der durch die neue Variable zu ersetzende Wert "PLExp" vom Typ "PrimitiveLiteralExp" gesucht (Z. 153). Im nächsten Schritt wird über den Aufruf der Hilfsmethode "getOperationCallExpression(allElements, PLExp)" die zugehörige Operation gesucht, die "PLExp" verwendet (Z. 154). Jetzt muss die zuvor verwendete "PrimitiveLiteralExp" durch die neue "varExp" ersetzt werden. Dazu wird überprüft, ob "PLExp" das Source-Element von "OpCExp" ist oder in deren Argumenten enthalten ist (Z. 155-168). Abhängig davon, wo die "PrimitiveLiteralExp" verwendet wurde, wird diese entweder direkt durch "varExp" ersetzt (Z. 155-162)oder mit Hilfe der Hilfsmethode "replaceArgument(arguments, PLExp, varExp)" in der Argumenten-Liste ersetzt(Z. 165-167). Am Ende erhält die Variable "var" den Wert von "PLExp" als "InitExpression" (Z. 169), während "letExp" die Variable "var" über "setVariable(var)" erhält (Z. 170). Damit ist die neue Variable in den OCL-Code integriert und das Refactoring abgeschlossen, was die Textausgabe der "Constraint" zeigt (Z. 171).

**Benutzte Hilfsmethoden:**

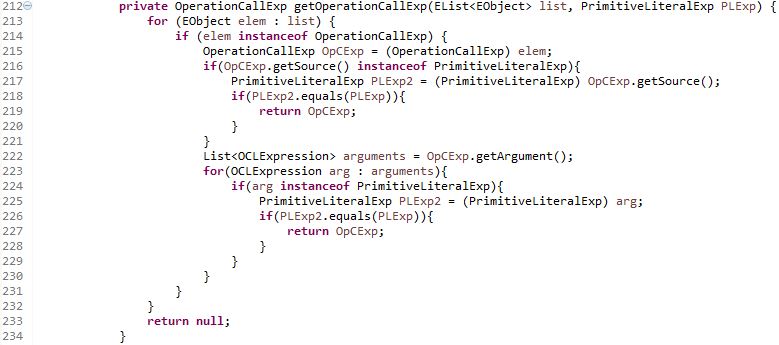
**getExpressionInOCL und getPrimitiveLiteralExp:**





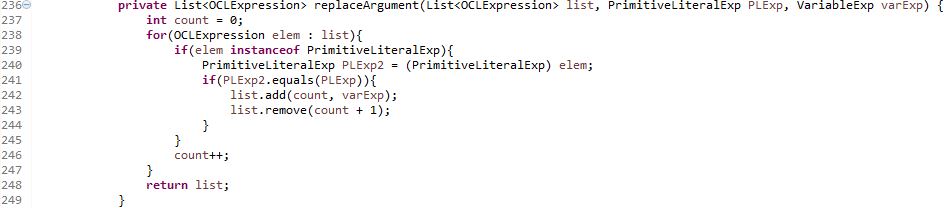
Beide Hilfsmethoden arbeiten nach dem gleichen Prinzip: Der Methode wird eine Liste übergeben, deren Elemente der Reihe nach durchlaufen werden (Z. 195, 204). Das erste Element vom gesuchten Typ wird als Ergebnis geliefert (Z. 196-197, 205-206).

**getOperationCallExp:**



Diese Hilfsmethode bekommt nicht nur eine Liste, sondern auch ein Element "PLExp" des Typs "PrimitiveLiteralExp" übergeben. Die Liste wird, ähnlich wie bei den vorherigen Hilfsmethoden, nach einer "OperationCallExp" durchsucht (Z. 213-214). Wird ein entsprechendes Element gefunden, wird überprüft, ob diese "OperationCallExp" das Element "PLExp" entweder als Source-Element (Z. 216-219) oder als Argument benutzt (Z. 222-227). Die erste "OperationCallExp", die diese Bedingung erfüllt, wird als Ergebnis zurückgegeben (Z. 219, 227).

**replaceArgument:**



Diese Hilfsmethode bekommt eine Liste, eine "PrimitiveLiteralExp" "PLExp" und eine "VariableExp" "varExp" übergeben. Die Liste wird anfangs nach dem Element "PLExp" durchsucht, wobei über die Variable "count" die Position in der Liste gespeichert wird (Z. 237, 246). Das entsprechende Element wird ermittelt, indem jedes gefundene Element des Typs "PrimitiveLiteralExp" mit "PLExp" verglichen wird (Z. 239-241). Wurde die Position von "PLExp" ermittelt, wird zuerst an dieser Position das Element "varExp" eingefügt (Z. 242). Danach wird das Element an der nächsten Position gelöscht (Z. 243), wobei es sich um das Element "PLExp" handelt, welches durch die vorherige Operation verschoben wurde. Damit wurde "PLExp" in der Liste durch "varExp" ersetzt, wobei die resultierende Liste als Ergebnis geliefert wird (Z. 248).

**Beispiel der Anwendung:**

Code vor dem Refactoring:

context Person invariant istFahrer: self.hatFahrerlaubnis.implies(self.alter.>=(17)).implies(self.fahrerlaubnis.ausstellungsjahr.<=(2015)).implies(self.strafpunkte.>=(0)).implies(self.strafpunkte.<(8))

Code nach dem Refactoring:

context Person invariant istFahrer: let test = 17 in self.hatFahrerlaubnis.implies(self.alter.>=(test)).implies(self.fahrerlaubnis.ausstellungsjahr.<=(2015)).implies(self.strafpunkte.>=(0)).implies(self.strafpunkte.<(8))

Durch das Refactoring wird die Variable "test" erstellt, wodurch der Ausdruck "let test = 17 in" vor dem ursprünglichen Code eingefügt wird. Im ursprünglichen Code wird der zu ersetzende Wert 17 durch die neue Variable "test" erstetzt.